|  |
| --- |
| TRƯỜNG ĐẠI HỌC SÀI GÒN  **KHOA CÔNG NGHỆ THÔNG TIN**  **CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**  Giảng viên hướng dẫn: Đỗ Như Tài Sinh viên thực hiện: Trần Thanh Sang MSSV:3124411257  Thành phố Hồ Chí Minh, tháng 03 năm 2025 |

Bài 1:

Main idea: Tạo node chứa thông tin của nút gốc, viết hàm tổng node, đếm node, trung bình cộng node để tính trung bình cộng các nút trong cây, hàm tổng node dương/âm, đếm node dương/âm, trung bình cộng node dương/âm, sau đó viết hàm tính tỷ số dương âm của các tổng số có nút có giá trị dương và giá trị âm.

Code:

#include<iostream>

using namespace std;

struct node

{

int info;

struct node\*left;

struct node\* right;

};

typedef struct node\* NODE;

typedef NODE TREE;

void init(TREE& Root)

{

Root = NULL;

}

void InsertNode(TREE& Root, int x)

{

NODE p = new node;

p->info = x;

p->left = p->right = NULL;

if (Root == NULL)

{

Root = p;

}

else

{

NODE q = Root;

NODE f = NULL;

while (q != NULL)

{

f = q;

if (q->info > x)

q = q->left;

else

q = q->right;

}

if (f->info > x)

f->left = p;

else

f->right = p;

}

}

int DemNode(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemNode(Root->left);

int b = DemNode(Root->right);

return a + b + 1;

}

int TongNode(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongNode(Root->left);

int b = TongNode(Root->right);

return a + b + Root->info;

}

float trungbingcong(TREE Root)

{

int s = TongNode(Root);

int dem = DemNode(Root);

if (dem==0)

return 0;

return (float)s / dem;

}

int DemDuong(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemDuong(Root->left);

int b = DemDuong(Root->right);

if (Root->info > 0)

return a + b + 1;

return a + b;

}

int TongDuong(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongDuong(Root->left);

int b = TongDuong(Root->right);

if (Root->info > 0)

return a + b + Root->info;

return a + b;

}

float TrungBinhDuong(TREE Root)

{

int s = TongDuong(Root);

int dem = DemDuong(Root);

if (dem == 0)

return 0;

return (float)s / dem;

}

int DemAm(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemAm(Root->left);

int b = DemAm(Root->right);

if (Root->info < 0)

return a + b + 1;

return a + b;

}

int TongAm(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongAm(Root->left);

int b = TongAm(Root->right);

if (Root->info < 0)

return a + b + Root->info;

return a + b;

}

float TrungBinhAm(TREE Root)

{

int s = TongAm(Root);

int dem = DemAm(Root);

if (dem == 0)

return 0;

return (float)s / dem;

}

float TinhTySo(TREE Root)

{

int a = DemDuong(Root);

int b = DemAm(Root);

if (b == 0)

return 0;

return (float)a / b;

}

void testcase(TREE& Root)

{

init(Root);

InsertNode(Root, 5);

InsertNode(Root, 3);

InsertNode(Root, 7);

InsertNode(Root, 2);

InsertNode(Root, 4);

InsertNode(Root, 6);

InsertNode(Root, 8);

InsertNode(Root, -1);

InsertNode(Root, 9);

cout << "Trung binh cong cac node: " << trungbingcong(Root) << endl;

cout << "Trung binh duong cac node: " << TrungBinhDuong(Root) << endl;

cout << "Trung binh am cac node: " << TrungBinhAm(Root) << endl;

cout << "Ty so duong am: " << TinhTySo(Root) << endl;

}

int main()

{

TREE Root;

testcase(Root);

return 0;

}

Testcase:  
  
Input: 5,3,7,2,4,6,8,-1,9  
Output:

Trung binh cong cac node: 4.77778

Trung binh duong cac node: 5.5

Trung binh am cac node: -1

Ty so duong am: 8

Bài 2:  
  
Main Idea: sử dụng biến nhớ phụ temp để hoán đổi vị trí của pleft, pright cũng như thay đổi cả hướng mà nó trỏ đến

Code:  
  
#include<iostream>

using namespace std;

struct node

{

int info;

struct node\*left;

struct node\* right;

};

typedef struct node\* NODE;

typedef NODE TREE;

void init(TREE& Root)

{

Root = NULL;

}

void InsertNode(TREE& Root, int x)

{

NODE p = new node;

p->info = x;

p->left = p->right = NULL;

if (Root == NULL)

{

Root = p;

}

else

{

NODE q = Root;

NODE f = NULL;

while (q != NULL)

{

f = q;

if (q->info > x)

q = q->left;

else

q = q->right;

}

if (f->info > x)

f->left = p;

else

f->right = p;

}

}

void PrintTree(TREE Root)

{

if (Root != NULL)

{

cout << Root->info << " ";

PrintTree(Root->left);

PrintTree(Root->right);

}

}

void DoiTraiPhai(TREE& Root)

{

if (Root == NULL) return;

NODE temp = Root->left;

Root->left = Root->right;

Root->right = temp;

temp = Root->left->left;

Root->left->left = Root->left->right;

Root->left->right = temp;

}

void testcase()

{

TREE Root;

init(Root);

int a[9] = { 10,5,15,3,9,12,18,7,20 };

cout << "Truoc khi hoan doi:\n";

for (int i = 0; i < 9; i++) {

InsertNode(Root, a[i]);

}

PrintTree(Root);

cout << endl;

DoiTraiPhai(Root);

cout << "Sau khi hoan doi (Print theo NLR):\n";

PrintTree(Root);

cout << endl;

}

int main()

{

testcase();

return 0;

}

Input: 10,5,15,3,9,12,18,7,20  
  
Output: 10 15 18 20 12 5 3 9 7 (print theo LNR)

Câu 3:

1. Thứ tự thêm: 10, 5, 15, 3, 9, 12, 18, 7, 20.
2. Là LNR.

Câu 4:

Main Idea: Viết hàm khởi tạo cây và hàm tìm nhỏ nhất và lớn nhất của các phần tử trong cây.

Code:

#include<iostream>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

}NODE;

typedef NODE\* TREE;

NODE\* NhoNhat(TREE Root)

{

if (Root == NULL)

return NULL;

NODE\* lc = Root;

while (lc->left != NULL)

lc = lc->left;

return lc;

}

NODE\* LonNhat(TREE Root)

{

if (Root == NULL)

return NULL;

NODE\* rc = Root;

while (rc->right != NULL)

rc = rc->right;

return rc;

}

void init(TREE& Root)

{

Root = NULL;

}

void insertNode(TREE& Root, int x)

{

if (Root == NULL)

{

NODE\* p = new NODE;

p->info = x;

p->left = NULL;

p->right = NULL;

Root = p;

}

else

{

if (Root->info == x)

return;

if (Root->info > x)

insertNode(Root->left, x);

else

insertNode(Root->right, x);

}

}

void testcase(TREE& Root)

{

int n;

cin >> n;

while (n--)

{

int x;

cin >> x;

insertNode(Root, x);

}

LonNhat(Root);

NhoNhat(Root);

cout << LonNhat(Root)->info << " " << NhoNhat(Root)->info << endl;

}

int main()

{

TREE Root;

init(Root);

testcase(Root);

return 0;

}

Testcase:  
Input: 10 5 3 9 7 15 12 18 20  
Output: 20 3

Câu 5:

Main Idea: Viết hàm tính tổng số nút có 1 nhánh con bằng cách đệ quy

Code:  
#include<iostream>

using namespace std;

typedef struct node

{

int data;

struct node\* left;

struct node\* right;

} NODE;

typedef NODE\* TREE;

int DemMotCon(TREE t)

{

if (t == NULL)

return 0;

if ((t->left && !t->right) || (!t->left && t->right))

return 1 + DemMotCon(t->left) + DemMotCon(t->right);

return DemMotCon(t->left) + DemMotCon(t->right);

}

Test case:

Nhập n=4:  
5,1,4,7  
  
Output: 1

Câu 6:  
Main idea: Tạo các hàm cài đặt cây cơ bản và tạo hàm đếm node và tổng các node

Code:

#include<iostream>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

}NODE;

typedef NODE\* TREE;

void init(TREE& Root)

{

Root = NULL;

}

void insertNode(TREE& Root, int x)

{

if (Root == NULL)

{

NODE\* p = new NODE;

p->info = x;

p->left = NULL;

p->right = NULL;

Root = p;

}

else

{

if (Root->info == x)

return;

if (Root->info > x)

insertNode(Root->left, x);

else

insertNode(Root->right, x);

}

int DemNode(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemNode(Root->left);

int b = DemNode(Root->right);

return a + b + 1;

}

int TongNode(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongNode(Root->left);

int b = TongNode(Root->right);

return a + b + Root->info;

}

void testcase(TREE& Root)

{

init(Root);

int n;

cin >> n;

for (int i = 1; i <= n; i++)

{

int x;

cin >> x;

insertNode(Root, x);

}

cout << DemNode(Root) << endl;

cout << TongNode(Root) << endl;

}

int main()

{

TREE Root;

testcase(Root);

return 0;

}

Testcase:  
Input: n[4]={5,1,4,7}

Output: 17

Câu 7:

Hãy mô tả ngắn gọn sự giống và khác nhau giữa

hai cấu trúc Cây Nhị Phân tìm kiếm và Danh Sách

Liên Kết Đơn?

Trả lời:

Giống nhau:

• Cùng là CTDL động.

• Các thao tác cơ bản Thêm, Xóa, Cập Nhật được thực hiện một

cách linh hoạt.

Khác nhau:

• Dữ liệu trên cây NPTK được tổ chức và DSLK đơn thì không.

• Chi phí tìm kiếm, thêm trên cây nhanh hơn trên DSLK đơn.

Câu 8:

Main idea: Cài đặt cây và sau đó viết hàm LNR, sau đó hàm xuất

Code:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<cstdio>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

}NODE;

typedef NODE\* TREE;

void LNR(TREE Root, FILE\* fp)

{

if (Root != NULL)

{

LNR(Root->left, fp);

fwrite(&Root->info, sizeof(int), 1, fp);

LNR(Root->right, fp);

}

}

int xuat(const char\* filename, TREE Root)

{

FILE\* fp = fopen(filename, "wb");

if (fp == NULL)

return 0;

LNR(Root, fp);

fclose(fp);

return 1;

}

void init(TREE& Root)

{

Root = NULL;

}

void insertNode(TREE& Root, int x)

{

if (Root == NULL)

{

NODE\* p = new NODE;

p->info = x;

p->left = NULL;

p->right = NULL;

Root = p;

}

else

{

if (Root->info == x)

return;

if (Root->info > x)

insertNode(Root->left, x);

else

insertNode(Root->right, x);

}

}

void testcase(TREE& Root)

{

init(Root);

int n;

cout << "Nhap so luong phan tu cua cay: ";

cin >> n;

for (int i = 1; i <= n; i++)

{

int x;

cout << "Nhap phan tu thu " << i << ": ";

cin >> x;

insertNode(Root, x);

}

xuat("output", Root);

}

int main()

{

TREE Root;

testcase(Root);

return 0;

}  
Test case:

Input: n[4]={5,1,3,7}

Output: Được đẩy ra file output với nội dung là 1 3 5 7

Câu 9:

Main Idea: Cài đặt cây và sau đó hàm NLR và hàm xuất trả kết quả

Code:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<cstdio>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

}NODE;

typedef NODE\* TREE;

void NLR(TREE Root, FILE\* fp)

{

if (Root != NULL)

{

fprintf(fp, "%d ", Root->info);

NLR(Root->left, fp);

NLR(Root->right, fp);

}

}

int xuat(const char\* filename, TREE Root)

{

FILE\* fp = fopen(filename, "w");

if (fp == NULL)

return 0;

NLR(Root, fp);

fclose(fp);

return 1;

}

void init(TREE& Root)

{

Root = NULL;

}

void insertNode(TREE& Root, int x)

{

if (Root == NULL)

{

NODE\* p = new NODE;

p->info = x;

p->left = NULL;

p->right = NULL;

Root = p;

}

else

{

if (Root->info == x)

return;

if (Root->info > x)

insertNode(Root->left, x);

else

insertNode(Root->right, x);

}

}

void testcase(TREE& Root)

{

init(Root);

int n;

cout << "Nhap so luong phan tu cua cay: ";

cin >> n;

for (int i = 1; i <= n; i++)

{

int x;

cout << "Nhap phan tu thu " << i << ": ";

cin >> x;

insertNode(Root, x);

}

xuat("output.txt", Root);

}

int main()

{

TREE Root;

testcase(Root);

return 0;

}

Testcase:

Input: n[4]={5,1,3,7}  
Output: Trả ra file có nội dung 5,1,3,7

Câu 10:  
Main Idea: Cài đặt cây và sau đó cày hàm LRN và hàm xuất trả kết quả

Code:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<cstdio>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

}NODE;

typedef NODE\* TREE;

void LRN(TREE Root, FILE\* fp)

{

if (Root != NULL)

{

LRN(Root->left, fp);

LRN(Root->right, fp);

fprintf(fp, "%d ", Root->info);

}

}

int xuat(const char\* filename, TREE Root)

{

FILE\* fp = fopen(filename, "w");

if (fp == NULL)

return 0;

LRN(Root, fp);

fclose(fp);

return 1;

}

void init(TREE& Root)

{

Root = NULL;

}

void insertNode(TREE& Root, int x)

{

if (Root == NULL)

{

NODE\* p = new NODE;

p->info = x;

p->left = NULL;

p->right = NULL;

Root = p;

}

else

{

if (Root->info == x)

return;

if (Root->info > x)

insertNode(Root->left, x);

else

insertNode(Root->right, x);

}

}

void testcase(TREE& Root)

{

init(Root);

int n;

cout << "Nhap so luong phan tu cua cay: ";

cin >> n;

for (int i = 1; i <= n; i++)

{

int x;

cout << "Nhap phan tu thu " << i << ": ";

cin >> x;

insertNode(Root, x);

}

xuat("output.txt", Root);

}

int main()

{

TREE Root;

testcase(Root);

return 0;

}

Testcase:

Input: n[4]={5,1,3,7}

Output: Trả file output với nội dung là 3,1,7,5

Câu 11:

Cho cây nhị phân tìm kiếm các số nguyên t. Hãy cho biết cách thức duyệt cây như thế nào để ta được thứ tự các giá trị tăng dần?

Trả lời: Sử dụng phép duyệt LNR

Câu 12:

Cho cây nhị phân tìm kiếm các số nguyên t. Hãy cho biết cách thức duyệt cây như thế nào để ta được thứ tự các giá trị giảm dần?

Trả Lời: Sử dụng phép duyệt RNL

Câu 13:

Main Idea: Cài đặt cây và sau đó hàm NLR và hàm Xuất để trả kết quả

Code:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<cstdio>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

}NODE;

typedef NODE\* TREE;

void NLR(TREE Root, FILE\* fp)

{

if (Root != NULL)

{

fprintf(fp, "%d ", Root->info);

NLR(Root->left, fp);

NLR(Root->right, fp);

}

}

int xuat(const char\* filename, TREE Root)

{

FILE\* fp = fopen(filename, "w");

if (fp == NULL)

return 0;

NLR(Root, fp);

fclose(fp);

return 1;

}

void init(TREE& Root)

{

Root = NULL;

}

void insertNode(TREE& Root, int x)

{

if (Root == NULL)

{

NODE\* p = new NODE;

p->info = x;

p->left = NULL;

p->right = NULL;

Root = p;

}

else

{

if (Root->info == x)

return;

if (Root->info > x)

insertNode(Root->left, x);

else

insertNode(Root->right, x);

}

}

void testcase(TREE& Root)

{

init(Root);

int n;

cout << "Nhap so luong phan tu cua cay: ";

cin >> n;

for (int i = 1; i <= n; i++)

{

int x;

cout << "Nhap phan tu thu " << i << ": ";

cin >> x;

insertNode(Root, x);

}

xuat("output.txt", Root);

}

int main()

{

TREE Root;

testcase(Root);

return 0;

}

Testcase:

Input: n[4]={5,1,3,7}  
Output: Trả ra file có nội dung 5,1,3,7

Câu 14:

Main Idea: Cài đặt cây , cài đặt danh sách liên kết đơn đơn theo cây, hàm khởi tạo, hàm nhập cây và hàm them đuôi, hàm build list đẻ tạo danh sách liên kết, hàm RNL.

Code:

#include <iostream>

using namespace std;

typedef struct node {

int info;

struct node\* left;

struct node\* right;

} Node;

typedef Node\* Tree;

typedef struct nodelist {

Node\* node;

struct nodelist\* next;

} nodelist;

typedef struct list {

nodelist\* head;

nodelist\* tail;

} List;

void init(List& l) {

l.head = l.tail = NULL;

}

void insertnode(Tree& Root, int x) {

if (Root == NULL) {

Node\* p = new Node;

p->info = x;

p->left = p->right = NULL;

Root = p;

}

else {

if (x < Root->info) {

insertnode(Root->left, x);

}

else if (x > Root->info) {

insertnode(Root->right, x);

}

}

}

nodelist\* getnode(Node\* x) {

nodelist\* p = new nodelist;

if (p == NULL) {

return NULL;

}

p->node = x;

p->next = NULL;

return p;

}

void addtail(List& l, nodelist\* p) {

if (l.head == NULL) {

l.head = l.tail = p;

}

else {

l.tail->next = p;

l.tail = p;

}

}

void RNL(Tree Root, List& l);

void buildlist(Tree Root, List& l) {

init(l);

RNL(Root, l);

}

void RNL(Tree Root, List& l) {

if (Root != NULL) {

RNL(Root->right, l);

nodelist\* p = getnode(Root);

addtail(l, p);

RNL(Root->left, l);

}

}

void freeList(List& l) {

nodelist\* p = l.head;

while (p != NULL) {

nodelist\* temp = p;

p = p->next;

delete temp;

}

l.head = l.tail = NULL;

}

void freeTree(Tree& Root) {

if (Root != NULL) {

freeTree(Root->left);

freeTree(Root->right);

delete Root;

Root = NULL;

}

}

void test(Tree Root) {

List l;

buildlist(Root, l);

nodelist\* p = l.head;

while (p != NULL) {

cout << p->node->info << " ";

p = p->next;

}

freeList(l);

}

int main() {

Tree Root = NULL;

insertnode(Root, 5);

insertnode(Root, 1);

insertnode(Root, 3);

insertnode(Root, 7);

test(Root);

return 0;

}

Test case:

Input: 5,1,3,7

Output: 7 5 3 1

Câu 15:

Giữa cấu trúc cây nhị phân tìm kiếm và cấu trúc mảng các phần tử được sắp thứ tự tăng dần có những điểm giống và khác nhau như thế nào?

Trả lời:

Giống nhau

• Dữ liệu được tổ chức.

• Chi phí tìm kiếm một phần tử trên cả hai ctdl là như nhau.

Khác nhau:

• Chi phí thêm và xoá phần tử vào mảng lớn hơn chi phí cây nhị

phân tìm kiếm.

Câu 16:

Main Idea: Cài đặt cây nhị phân tìm kiếm như yêu cầu đề, cài hàm deletenode, hàm xoá giá trị hàm liệt kê và hàm NLR

Code:

#include<iostream>

using namespace std;

struct BST\_NODE

{

int Key;

int So\_lan;

struct BST\_NODE\* Left, \* Right;

};

struct BST\_TREE

{

BST\_NODE\* pRoot;

};

struct BST\_TREE t;

int deleteNode(BST\_NODE\*& root, int x) {

if (root == NULL) return 0;

if (x < root->Key) {

return deleteNode(root->Left, x);

}

else if (x > root->Key) {

return deleteNode(root->Right, x);

}

else {

if (root->So\_lan > 1) {

root->So\_lan--;

return 1;

}

if (root->Left == NULL) {

BST\_NODE\* temp = root->Right;

delete root;

root = temp;

}

else if (root->Right == NULL) {

BST\_NODE\* temp = root->Left;

delete root;

root = temp;

}

else {

BST\_NODE\* temp = root->Right;

while (temp->Left != NULL) {

temp = temp->Left;

}

root->Key = temp->Key;

root->So\_lan = temp->So\_lan;

deleteNode(root->Right, temp->Key);

}

return 1;

}

}

void insertNode(BST\_NODE\*& root, int x) {

if (root == NULL) {

root = new BST\_NODE;

root->Key = x;

root->So\_lan = 1;

root->Left = root->Right = NULL;

}

else if (x == root->Key) {

root->So\_lan++;

}

else if (x < root->Key) {

insertNode(root->Left, x);

}

else {

insertNode(root->Right, x);

}

}

void XoaGiaTri(struct BST\_TREE& t, int x)

{

if (deleteNode(t.pRoot, x) == 0)

{

cout << "Khong tim thay gia tri can xoa" << endl;

}

else

{

cout << "Xoa thanh cong" << endl;

}

}

void init(struct BST\_TREE& t)

{

t.pRoot = NULL;

}

void insertNode(struct BST\_TREE& t, int x)

{

if (t.pRoot == NULL)

{

t.pRoot = new BST\_NODE;

t.pRoot->Key = x;

t.pRoot->So\_lan = 1;

t.pRoot->Left = NULL;

t.pRoot->Right = NULL;

}

else

{

if (t.pRoot->Key == x)

{

t.pRoot->So\_lan++;

}

else if (x < t.pRoot->Key)

{

t.pRoot = t.pRoot->Left;

insertNode(t, x);

}

else

{

t.pRoot = t.pRoot->Right;

insertNode(t, x);

}

}

}

void Xuat(BST\_NODE\* root)

{

if (root != NULL)

{

Xuat(root->Left);

cout << root->Key << " " << root->So\_lan << endl;

Xuat(root->Right);

}

}

void lietke(struct BST\_TREE& t)

{

Xuat(t.pRoot);

}

void NLR(struct BST\_NODE\* root)

{

if (root != NULL)

{

cout << root->Key << " " << root->So\_lan << endl;

NLR(root->Left);

NLR(root->Right);

}

}

void XuatNLR(struct BST\_TREE& t)

{

NLR(t.pRoot);

}

void testcase1() {

BST\_TREE t;

init(t);

insertNode(t.pRoot, 5);

insertNode(t.pRoot, 1);

insertNode(t.pRoot, 3);

insertNode(t.pRoot, 7);

cout << "Truoc khi xoa:" << endl;

Xuat(t.pRoot);

XoaGiaTri(t, 7);

cout << "Sau khi xoa:" << endl;

Xuat(t.pRoot);

}

int main()

{

testcase1();

return 0;

}

Testcase:

Input: 5,3,1,7

Xoá giá trị 7

Output: 1,3,5

Câu 001:

Main Idea: Cài đặt cây, và cài đặt hàm xoá hết tất cả phần tử\

Code:

#include<iostream>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

} Node;

typedef Node\* Tree;

void init(Tree& t)

{

t = NULL;

}

void insert(Tree& t, int x)

{

if (t == NULL)

{

t = new Node;

t->info = x;

t->left = NULL;

t->right = NULL;

}

else

{

if (t->info > x)

insert(t->left, x);

else if (t->info < x)

insert(t->right, x);

}

}

void Xuat(Tree t)

{

if (t != NULL)

{

Xuat(t->left);

cout << t->info << " ";

Xuat(t->right);

}

}

void removeall(Tree& t)

{

if (t == NULL)

return;

removeall(t->left);

removeall(t->right);

delete t;

t = NULL;

}

void test()

{

Tree t;

init(t);

insert(t, 5);

insert(t, 1);

insert(t, 3);

insert(t, 7);

Xuat(t);

cout << endl;

removeall(t);

if (t == NULL)

cout << "Xoa thanh cong";

else

cout << "Xoa that bai";

}

int main()

{

test();

return 0;

}

Testcase:

Input: 5,1,3,7

Output: Xoá Thành Công

Câu 17:  
  
Hàm tìm phần tử tận cùng bên trái của nhánh phải p:

void searchstandfor(TREE& p, TREE& q)

{

if (q->left)

{

searchstandfor(p, q->left);

}

else

{

p->info = q->info;

p = q;

q = q->right;

}

}

Câu 18:

Câu a: Là LNR

Câu b và c:

#include<iostream>

using namespace std;

typedef struct node

{

int info;

struct node\* left;

struct node\* right;

} Node;

typedef Node\* Tree;

void init(Tree& t)

{

t = NULL;

}

Node\* GetNode(int x)

{

Node\* p = new Node;

if (p == NULL)

return NULL;

p->info = x;

p->left = p->right = NULL;

return p;

}

int insert(Tree& t, int x)

{

if (t)

{

if (t->info == x)

return 0;

if (t->info > x)

return insert(t->left, x);

else

return insert(t->right, x);

}

t = GetNode(x);

if (t == NULL)

return -1;

return 1;

}

int TaoCay(Tree& t, int a[], int n)

{

init(t);

for (int i = 0; i < n; i++)

{

if (insert(t, a[i]) == -1)

return 0;

}

return 1;

}

void LNR(Tree t, int a[], int& n)

{

if (t == NULL)

return;

LNR(t->left, a, n);

a[n++] = t->info;

LNR(t->right, a, n);

}

void Xuat(Tree t, int a[], int& n)

{

n = 0;

LNR(t, a, n);

for (int i = 0; i < n; i++)

{

cout << a[i] << " ";

}

}

void test()

{

Tree t;

init(t);

int a[] = { 5, 1, 3, 7 };

int n = 4;

TaoCay(t, a, n);

int b[100];

int count = 0;

Xuat(t, b, count);

}

int main()

{

test();

return 0;

}

Testcase:

Input: a[4]={5,1,3,7}

Output: 1 3 5 7

Câu 33:  
  
Dùng Stack:

void dungStack(Node\* root) {

if (root == NULL) return;

stack<Node\*> s;

s.push(root);

while (!s.empty()) {

Node\* current = s.top();

s.pop();

cout << current->data << " ";

if (current->right != NULL)

s.push(current->right);

if (current->left != NULL)

s.push(current->left);

}

}

Dùng Queue:

void dungQueue(Node\* root) {

if (root == NULL) return;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* current = q.front();

q.pop();

cout << current->data << " ";

if (current->left != NULL)

q.push(current->left);

if (current->right != NULL)

q.push(current->right);

}

}

Câu 37:

Main Idea: Cài đặt cây với init và insert, cài đặt hàm lớn nhất và nhỏ nhất và hàm tính chiều cao của cây

Code:

#include <iostream>

using namespace std;

typedef struct node {

int info;

struct node\* left;

struct node\* right;

}Node;

typedef Node\* Tree;

void init(Tree& t) {

t = NULL;

}

Node\* createNode(int x) {

Node\* p = new Node;

if (!p) {

exit(1);

}

p->info = x;

p->left = p->right = NULL;

return p;

}

Node\* LonNhat(Tree t)

{

if (t == NULL) {

return NULL;

}

Node\* lc = t;

Node\* a = LonNhat(t->left);

if (a && a->info > lc->info)

{

lc = a;

}

Node\* b = LonNhat(t->right);

if (b && b->info > lc->info)

{

lc = b;

}

return lc;

}

Node\* NhoNhat(Tree t)

{

if (t == NULL) {

return NULL;

}

Node\* lc = t;

Node\* a = NhoNhat(t->left);

if (a && a->info < lc->info)

{

lc = a;

}

Node\* b = NhoNhat(t->right);

if (b && b->info < lc->info)

{

lc = b;

}

return lc;

}

int ChieuCao(Tree t)

{

if (t == NULL)

{

return 0;

}

int a = ChieuCao(t->left);

int b = ChieuCao(t->right);

if (a > b)

{

return a + 1;

}

return b + 1;

}

int ktCanBang(Tree t)

{

if (t == NULL)

{

return 1;

}

if (ktCanBang(t->left) == 0)

{

return 0;

}

if (ktCanBang(t->right) == 0)

{

return 0;

}

Node\* a = LonNhat(t->left);

if (a && a->info > t->info)

{

return 0;

}

a = NhoNhat(t->right);

if (a && a->info < t->info)

{

return 0;

}

int x = ChieuCao(t->left);

int y = ChieuCao(t->right);

if (abs(x - y) > 1)

{

return 0;

}

return 1;

}

void insertNode(Tree& t, int x) {

Node\* p = createNode(x);

if (t == NULL) {

t = p;

}

else {

Node\* f = NULL;

Node\* q = t;

while (q != NULL) {

f = q;

if (q->info == x) {

return;

}

else if (q->info > x) {

q = q->left;

}

else {

q = q->right;

}

}

if (f->info > x) {

f->left = p;

}

else {

f->right = p;

}

}

}

void xuat(Tree t) {

if (t != NULL) {

cout << t->info << " ";

xuat(t->left);

xuat(t->right);

}

}

void test()

{

Tree t;

init(t);

insertNode(t, 5);

insertNode(t, 1);

insertNode(t, 3);

insertNode(t, 7);

ktCanBang(t);

xuat(t);

cout << "\nCay co can bang khong? " << (ktCanBang(t) ? "Co" : "Khong") << endl;

}

int main() {

test();

return 0;

}

Testcase:

Input: a[4]={5,1,3,7}

Output: Cay Co Can Bang